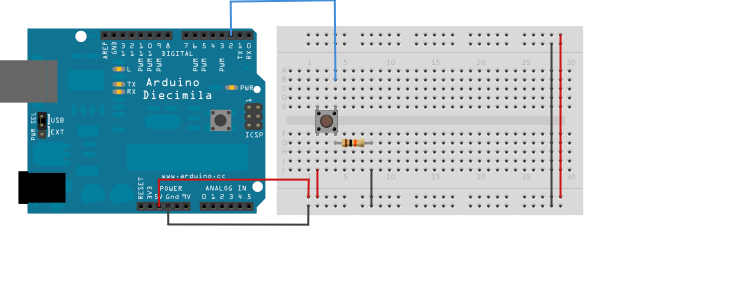
**Lab 6 – Buttons! (also known as switches)**

***The lab introduces the digital input features of the microcontroller. The Arduino’s output pins are labeled 0-13 (across from the Power and Analog In). Connecting a switch to a pin can activate a line in your program, and ultimately allow a user to control an output.***

**Hardware Required**Arduino Board LED  
10kΩ resistor (10,000 ohms)  
pushbutton switch  
multimeter (continuity function)   
miscellaneous jumper wires

**Circuit (set up with power off)**

Connect three wires to the Arduino board (pin2, 5V, Gnd). The first two, red and black, connect to the two long vertical rows on the side of the breadboard to provide access to the 5 volt supply and ground. The third wire goes from digital pin 2 to one leg of the pushbutton. That same leg of the button connects through a pull-down resistor (10kΩ) to ground. The other leg of the button connects to the 5 volt supply. Check that the pushbutton works with the multimeter before connecting power to the circuit. Don’t forget to connect an LED on pin 13, and Gnd.

When the pushbutton is open (not pressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the “pull-down” resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.

“*Pull-up resistors are used in electronic logic circuits to ensure that inputs to logic systems settle at expected logic levels if external devices are disconnected or high-impedance.” (wikipedi,2010)*

You could also wire this circuit the opposite way, with a ‘pull-up’ resistor keeping the input HIGH, and going LOW when the button is pressed. If so, the behavior of the sketch will be reversed, with the LED normally on and turning off when you press the button.

*The information for this lab was borrowed from http://www.arduino.cc/en/Tutorial/Button*

**Basic Code Explanation**

In the program below, only a few lines of code are used to obtain the pushbutton’s state. Using that information, the program either leaves the light off, or turns it on.

**Directions**

Copy and run the sample code(s). You may ignore the comments, but they are vital for understanding the programming. Don’t ignore their importance when you’re copying/running/ and modifying the code!   
  
HINT: make sure the LED works by testing the blink code first.

Sample code 1

------------------------------------------------------------------

*/\* Turns on and off a light emitting diode(LED) connected to digital*

*pin 13, when pressing a pushbutton attached to pin 2.*

*This example code is in the public domain. \*/*

*const int buttonPin = 2; // constants won't change. They're used here to*

*// set pin numbers:*

*// the number of the pushbutton pin.*

*const int ledPin = 13; // the number of the LED pin.*

*// variables will change:*

*int buttonState = 0; // variable for reading the pushbutton status*

*void setup() {*

*pinMode(ledPin, OUTPUT);*

*pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input:*

*}*

*void loop(){*

*buttonState = digitalRead(buttonPin); // read the state of the pushbutton*

*// value:*

*// check if the pushbutton is pressed.*

*if (buttonState == HIGH) { // if it is, the buttonState is HIGH:*

*digitalWrite(ledPin, HIGH); // turn LED on:*

*}*

*else {*

*digitalWrite(ledPin, LOW); // turn LED off:*

*}*

*}*

------------------------------------------------------------------

Once the program works, observe its behavior. Obtain the instructor’s initials here (\_\_\_).

When you press the button, what happens?

When you release the button why doesn’t the LED stay on?

If you press the button for a longer period of time, what happens?

Put a delay in the program to allow the LED to stay on for a predetermined amount of time. (\_\_\_)

Sample code 2

------------------------------------------------------------------

*#define LED 13 // the pin for the LED   
#define BUTTON 2 // the input pin where the pushbutton is connected*

*int val = 0; // val will be used to store the state of the input pin   
int state = 0; // 0 = LED off while 1 = LED on*

*void setup() {   
 pinMode(LED, OUTPUT); // tell Arduino LED is an output   
 pinMode(BUTTON, INPUT); // and BUTTON is an input   
}*

*void loop() {   
 val = digitalRead(BUTTON); // read input value and store it*

*if (val == HIGH) { // check if the input is HIGH (button pressed)   
 state = 1 - state; // and change the state  
 }*

*if (state == 1) {   
 digitalWrite(LED, HIGH); // turn LED ON   
 }   
else {   
 digitalWrite(LED, LOW);   
 }   
}*

------------------------------------------------------------------

Once the program works, observe its behavior. Obtain the instructor’s initials here (\_\_\_).

**Modify the Program**Press and hold the button, and observe the program behavior (\_\_\_). What would account for this?  
Search the Arduino site or the internet for “debounce” and define it here.

How do you avoid the problems with debouncing?

Add short delays within your program (\_\_\_). Summarize your findings.

**\*Bonus\*** (+5) ---------------------------------------------------------------------------------

Create a program to display the pushbutton values (HIGH or LOW) on the serial monitor, while the program controls the light. (\_\_\_)